NXP USA Inc. - MC68HC000EI16R2 Datasheet

Welcome to E-XFL.COM

Understanding Embedded - Microprocessors

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of Embedded - Microprocessors

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details	
Product Status	Obsolete
Core Processor	EC000
Number of Cores/Bus Width	1 Core, 32-Bit
Speed	16MHz
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	5.0V
Operating Temperature	0°C ~ 70°C (TA)
Security Features	-
Package / Case	68-LCC (J-Lead)
Supplier Device Package	68-PLCC (24.21x24.21)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc000ei16r2

TABLE OF CONTENTS (Continued)

When a data register is used as either a source or a destination operand, only the appropriate low-order portion is changed; the remaining high-order portion is neither used nor changed.

2.3.2 Address Registers

Each address register (and the stack pointer) is 32 bits wide and holds a full, 32-bit address. Address registers do not support byte-sized operands. Therefore, when an address register is used as a source operand, either the low-order word or the entire long-word operand is used, depending upon the operation size. When an address register is used as the destination operand, the entire register is affected, regardless of the operation size. If the operation size is word, operands are sign-extended to 32 bits before the operation is performed.

2.4 DATA ORGANIZATION IN MEMORY

Bytes are individually addressable. As shown in Figure 2-5, the high-order byte of a word has the same address as the word. The low-order byte has an odd address, one count higher. Instructions and multibyte data are accessed only on word (even byte) boundaries. If a long-word operand is located at address n (n even), then the second word of that operand is located at address n+2.

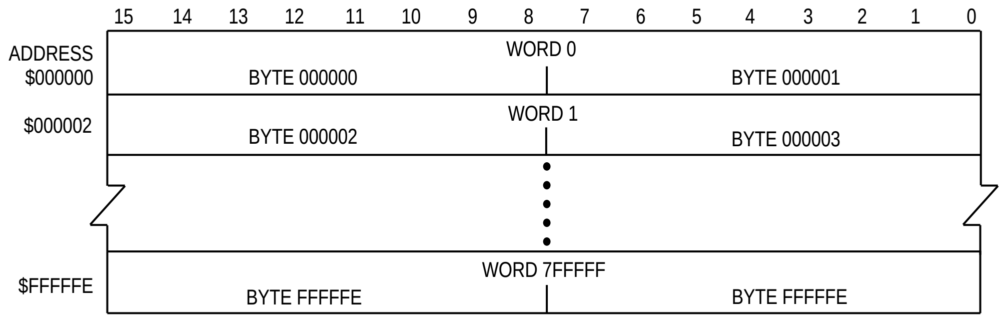

Figure 2-5. Word Organization in Memory

The data types supported by the M68000 MPUs are bit data, integer data of 8, 16, and 32 bits, 32-bit addresses, and binary-coded-decimal data. Each data type is stored in memory as shown in Figure 2-6. The numbers indicate the order of accessing the data from the processor. For the MC68008 with its 8-bit bus, the appearance of data in memory is identical to the all the M68000 MPUs. The organization of data in the memory of the MC68008 is shown in Figure 2-7.

Notation for operands:

PC —	Program counter
SR —	Status register
V —	Overflow condition code
Immediate Data —	Immediate data from the instruction
Source —	Source contents
Destination —	Destination contents
Vector —	Location of exception vector
+inf —	Positive infinity
–inf —	Negative infinity
<fmt> —	Operand data format: byte (B), word (W), long (L), single (S), double (D), extended (X), or packed (P).
FPm —	One of eight floating-point data registers (always specifies the source register)
FPn —	One of eight floating-point data registers (always specifies the destination register)

Notation for subfields and qualifiers:

<bit> of <operand> —	Selects a single bit of the operand
<ea>{offset:width} —	Selects a bit field
(<operand>) —	The contents of the referenced location
<operand>10 —	The operand is binary-coded decimal, operations are performed in decimal
(<address register>) —	The register indirect operator
–(<address register>) —	Indicates that the operand register points to the memory
(<address register>)+ —	Location of the instruction operand—the optional mode qualifiers are –, +, (d), and (d, ix)
#xxx or #<data> —	Immediate data that follows the instruction word(s)

Notations for operations that have two operands, written <operand> <op> <operand>, where <op> is one of the following:

→ —	The source operand is moved to the destination operand
↔ —	The two operands are exchanged
+ —	The operands are added
– —	The destination operand is subtracted from the source operand
× —	The operands are multiplied
÷ —	The source operand is divided by the destination operand
< —	Relational test, true if source operand is less than destination operand
> —	Relational test, true if source operand is greater than destination operand
V —	Logical OR
⊕ —	Logical exclusive OR
Λ —	Logical AND

Table 2-2. Instruction Set Summary (Sheet 4 of 4)

Opcode	Operation	Syntax

RTE	If supervisor state
	then (SP) $\rightarrow$ SR; SP + 2 $\rightarrow$ SP; (SP) $\rightarrow$ PC;
	SP + 4 $\rightarrow$ SP;
	restore state and deallocate stack according to (SP)
	else TRAP
		RTE

RTR	(SP) $\rightarrow$ CCR; SP + 2 $\rightarrow$ SP; (SP) $\rightarrow$ PC; SP + 4 $\rightarrow$ SP	RTR
RTS	(SP) $\rightarrow$ PC; SP + 4 $\rightarrow$ SP	RTS
SBCD	Destination_{10} – Source_{10} – X $\rightarrow$ Destination	SBCD Dx,Dy SBCD –(Ax),–(Ay)
Scc	If condition true then 1s $\rightarrow$ Destination else 0s $\rightarrow$ Destination	Scc <ea>
STOP	If supervisor state then Immediate Data $\rightarrow$ SR; STOP else TRAP	STOP # <data>
SUB	Destination – Source $\rightarrow$ Destination	SUB <ea>,Dn SUB Dn,<ea>
SUBA	Destination – Source $\rightarrow$ Destination	SUBA <ea>,An
SUBI	Destination – Immediate Data $\rightarrow$ Destination	SUBI # <data>,<ea>
SUBQ	Destination – Immediate Data $\rightarrow$ Destination	SUBQ # <data>,<ea>
SUBX	Destination – Source – X $\rightarrow$ Destination	SUBX Dx,Dy SUBX –(Ax),–(Ay)
SWAP	Register [31:16] $\leftrightarrow$ Register [15:0]	SWAP Dn
TAS	Destination Tested $\rightarrow$ Condition Codes; 1 $\rightarrow$ bit 7 of Destination	TAS <ea>

TRAP	SSP – 2 $\rightarrow$ SSP; Format/Offset $\rightarrow$ (SSP);
	SSP – 4 $\rightarrow$ SSP; PC $\rightarrow$ (SSP); SSP–2 $\rightarrow$ SSP;
	SR $\rightarrow$ (SSP); Vector Address $\rightarrow$ PC
		TRAP # <vector>

TRAPV	If V then TRAP	TRAPV
TST	Destination Tested $\rightarrow$ Condition Codes	TST <ea>
UNLK	An $\rightarrow$ SP; (SP) $\rightarrow$ An; SP + 4 $\rightarrow$ SP	UNLK An

NOTE: d is direction, L or R.

Address Bus (A23–A0)

This 24-bit, unidirectional, three-state bus is capable of addressing 16 Mbytes of data. This bus provides the address for bus operation during all cycles except interrupt acknowledge cycles and breakpoint cycles. During interrupt acknowledge cycles, address lines A1, A2, and A3 provide the level number of the interrupt being acknowledged, and address lines A23–A4 and A0 are driven to logic high. In 16-Bit mode, A0 is always driven high.

MC68008 Address Bus

The unidirectional, three-state buses in the two versions of the **MC68008** differ from each other and from the other processor bus only in the number of address lines and the addressing range. The 20-bit address (A19–A0) of the 48-pin version provides a 1-Mbyte address space; the 52-pin version supports a 22-bit address (A21–A0), extending the address space to 4 Mbytes. During an interrupt acknowledge cycle, the interrupt level number is placed on lines A1, A2, and A3. Lines A0 and A4 through the most significant address line are driven to logic high.

3.2 DATA BUS (D15–D0; MC68008: D7–D0)

This bidirectional, three-state bus is the general-purpose data path. It is 16 bits wide in the all the processors except the **MC68008** which is 8 bits wide. The bus can transfer and accept data of either word or byte length. During an interrupt acknowledge cycle, the external device supplies the vector number on data lines D7–D0. The MC68EC000 and MC68HC001 use D7–D0 in 8-bit mode, and D15–D8 are undefined.

3.3 ASYNCHRONOUS BUS CONTROL

Asynchronous data transfers are controlled by the following signals: address strobe, read/write, upper and lower data strobes, and data transfer acknowledge. These signals are described in the following paragraphs.

Address Strobe ($\overline{\text{AS}}$).

This three-state signal indicates that the information on the address bus is a valid address.

Read/Write (R/$\overline{\text{W}}$).

This three-state signal defines the data bus transfer as a read or write cycle. The R/$\overline{\text{W}}$ signal relates to the data strobe signals described in the following paragraphs.

Upper And Lower Data Strobes ($\overline{\text{UDS}}$, $\overline{\text{LDS}}$).

These three-state signals and R/$\overline{\text{W}}$ control the flow of data on the data bus. Table 3-1 lists the combinations of these signals and the corresponding data on the bus. When the R/$\overline{\text{W}}$ line is high, the processor reads from the data bus. When the R/$\overline{\text{W}}$ line is low, the processor drives the data bus. In 8-bit mode, $\overline{\text{UDS}}$ is always forced high and the $\overline{\text{LDS}}$ signal is used.

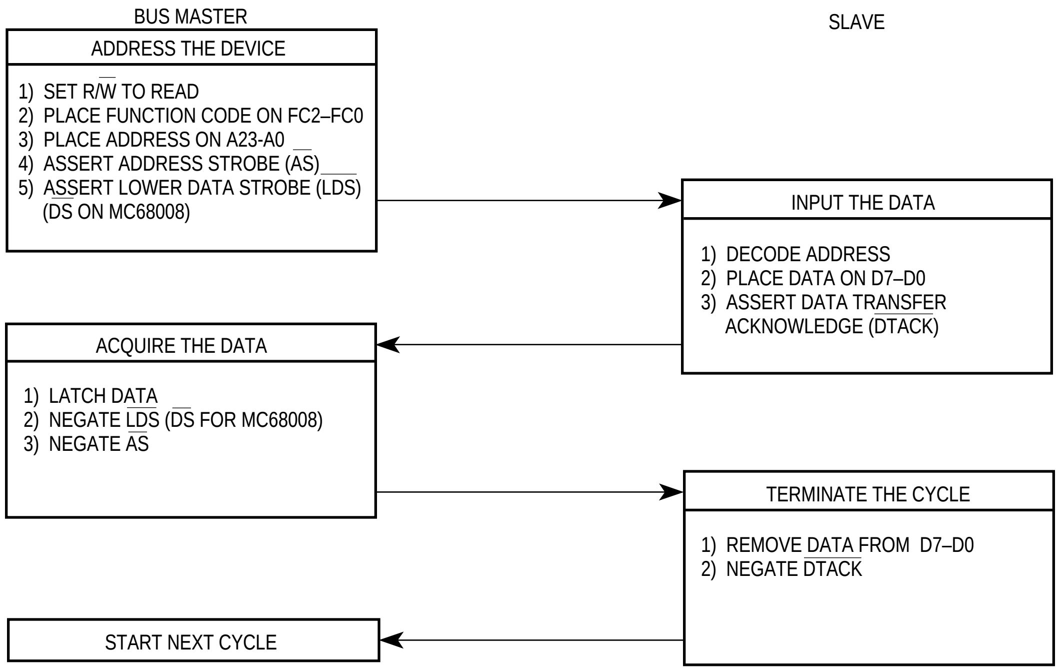
BUS MASTER
SLAVE
ADDRESS THE DEVICE
1) SET R/W TO READ
2) PLACE FUNCTION CODE ON FC2–FC0
3) PLACE ADDRESS ON A23-A0
4) ASSERT ADDRESS STROBE (AS)
5) ASSERT LOWER DATA STROBE (LDS) (DS ON MC68008)
INPUT THE DATA
1) DECODE ADDRESS
2) PLACE DATA ON D7–D0
3) ASSERT DATA TRANSFER ACKNOWLEDGE (DTACK)
ACQUIRE THE DATA
1) LATCH DATA
2) NEGATE LDS (DS FOR MC68008)
3) NEGATE AS
TERMINATE THE CYCLE
1) REMOVE DATA FROM D7–D0
2) NEGATE DTACK
START NEXT CYCLE

Figure 4-1. Byte Read-Cycle Flowchart

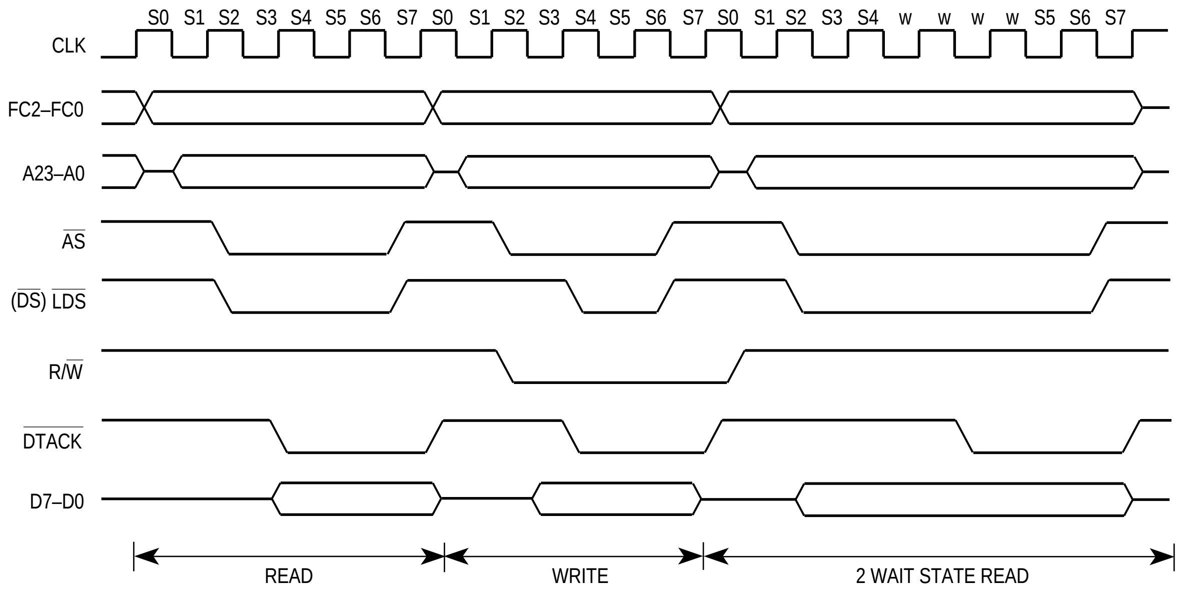
S0 S1 S2 S3 S4 S5 S6 S7 S0 S1 S2 S3 S4 S5 S6 S7 S0 S1 S2 S3 S4 w w w w S5 S6 S7
CLK
FC2–FC0
A23–A0
AS
(DS) LDS
R/W
DTACK
D7–D0
READ
WRITE
2 WAIT STATE READ

Figure 4-2. Read and Write-Cycle Timing Diagram

A bus cycle consists of eight states. The various signals are asserted during specific states of a read cycle, as follows:

STATE 0	The read cycle starts in state 0 (S0). The processor places valid function codes on FC0–FC2 and drives R/$\overline{\text{W}}$ high to identify a read cycle.
STATE 1	Entering state 1 (S1), the processor drives a valid address on the address bus.
STATE 2	On the rising edge of state 2 (S2), the processor asserts $\overline{\text{AS}}$ and $\overline{\text{UDS}}$, $\overline{\text{LDS}}$, or $\overline{\text{DS}}$.
STATE 3	During state 3 (S3), no bus signals are altered.
STATE 4	During state 4 (S4), the processor waits for a cycle termination signal ($\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$) or $\overline{\text{VPA}}$, an M6800 peripheral signal. When $\overline{\text{VPA}}$ is asserted during S4, the cycle becomes a peripheral cycle (refer to **Appendix B M6800 Peripheral Interface**). If neither termination signal is asserted before the falling edge at the end of S4, the processor inserts wait states (full clock cycles) until either $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ is asserted.
STATE 5	During state 5 (S5), no bus signals are altered.
STATE 6	During state 6 (S6), data from the device is driven onto the data bus.
STATE 7	On the falling edge of the clock entering state 7 (S7), the processor latches data from the addressed device and negates $\overline{\text{AS}}$, $\overline{\text{UDS}}$, and $\overline{\text{LDS}}$. At the rising edge of S7, the processor places the address bus in the high-impedance state. The device negates $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ at this time.

NOTE

During an active bus cycle, $\overline{\text{VPA}}$ and $\overline{\text{BERR}}$ are sampled on every falling edge of the clock beginning with S4, and data is latched on the falling edge of S6 during a read cycle. The bus cycle terminates in S7, except when $\overline{\text{BERR}}$ is asserted in the absence of $\overline{\text{DTACK}}$. In that case, the bus cycle terminates one clock cycle later in S9.

5.1.2 Write Cycle

During a write cycle, the processor sends bytes of data to the memory or peripheral device. If the instruction specifies a word operation, the processor issues both $\overline{\text{UDS}}$ and $\overline{\text{LDS}}$ and writes both bytes. When the instruction specifies a byte operation, the processor uses the internal A0 bit to determine which byte to write and issues the appropriate data strobe. When the A0 bit equals zero, $\overline{\text{UDS}}$ is asserted; when the A0 bit equals one, $\overline{\text{LDS}}$ is asserted.

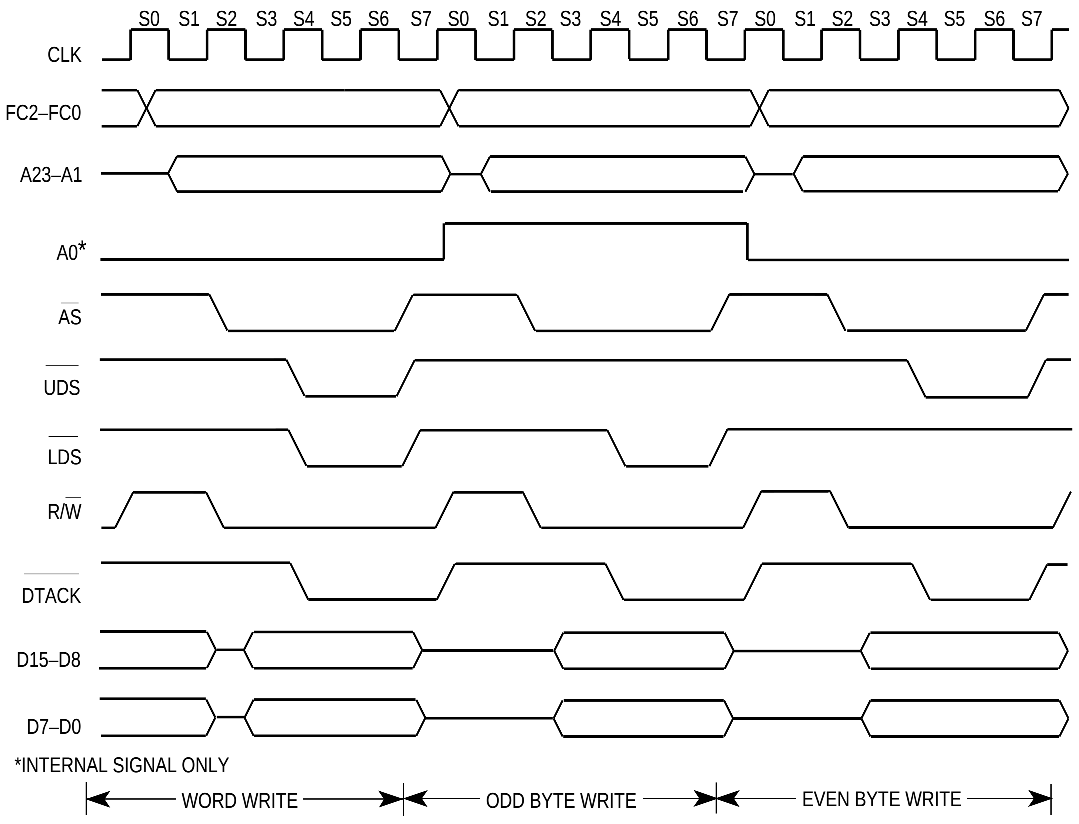

Figure 5-7. Word and Byte Write-Cycle Timing Diagram

The descriptions of the eight states of a write cycle are as follows:

STATE 0 The write cycle starts in S0. The processor places valid function codes on FC2–FC0 and drives R/$\overline{\text{W}}$ high (if a preceding write cycle has left R/$\overline{\text{W}}$ low).

STATE 1 Entering S1, the processor drives a valid address on the address bus.

STATE 2 On the rising edge of S2, the processor asserts $\overline{\text{AS}}$ and drives R/$\overline{\text{W}}$ low.

STATE 3 During S3, the data bus is driven out of the high-impedance state as the data to be written is placed on the bus.

STATE 4 At the rising edge of S4, the processor asserts $\overline{\text{UDS}}$, or $\overline{\text{LDS}}$. The processor waits for a cycle termination signal ($\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$) or $\overline{\text{VPA}}$, an M6800 peripheral signal. When $\overline{\text{VPA}}$ is asserted during S4, the cycle becomes a peripheral cycle (refer to **Appendix B M6800 Peripheral Interface**. If neither termination signal is asserted before the falling edge at the end of S4, the processor inserts wait states (full clock cycles) until either $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ is asserted.

STATE 5 During S5, no bus signals are altered.

STATE 6 During S6, no bus signals are altered.

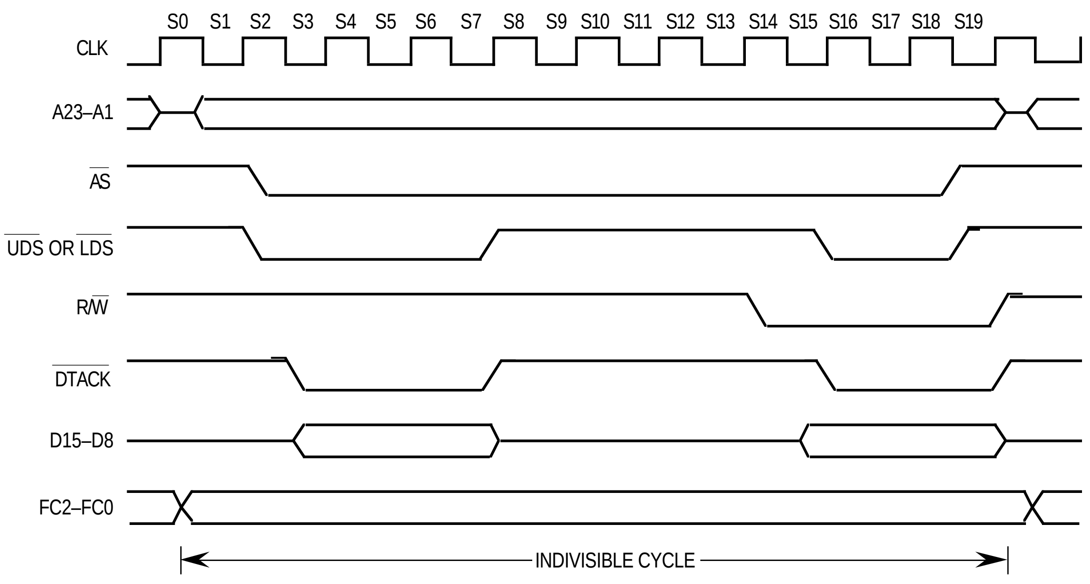

Figure 5-9. Read-Modify-Write Cycle Timing Diagram

The descriptions of the read-modify-write cycle states are as follows:

STATE 0 The read cycle starts in S0. The processor places valid function codes on FC2–FC0 and drives R/$\overline{\text{W}}$ high to identify a read cycle.

STATE 1 Entering S1, the processor drives a valid address on the address bus.

STATE 2 On the rising edge of S2, the processor asserts $\overline{\text{AS}}$ and $\overline{\text{UDS}}$, or $\overline{\text{LDS}}$.

STATE 3 During S3, no bus signals are altered.

STATE 4 During S4, the processor waits for a cycle termination signal ($\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$) or $\overline{\text{VPA}}$, an M6800 peripheral signal. When $\overline{\text{VPA}}$ is asserted during S4, the cycle becomes a peripheral cycle (refer to **Appendix B M6800 Peripheral Interface**). If neither termination signal is asserted before the falling edge at the end of S4, the processor inserts wait states (full clock cycles) until either $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ is asserted.

STATE 5 During S5, no bus signals are altered.

STATE 6 During S6, data from the device are driven onto the data bus.

STATE 7 On the falling edge of the clock entering S7, the processor accepts data from the device and negates $\overline{\text{UDS}}$, and $\overline{\text{LDS}}$. The device negates $\overline{\text{DTACK}}$ or $\overline{\text{BERR}}$ at this time.

STATES 8–11

The bus signals are unaltered during S8–S11, during which the arithmetic logic unit makes appropriate modifications to the data.

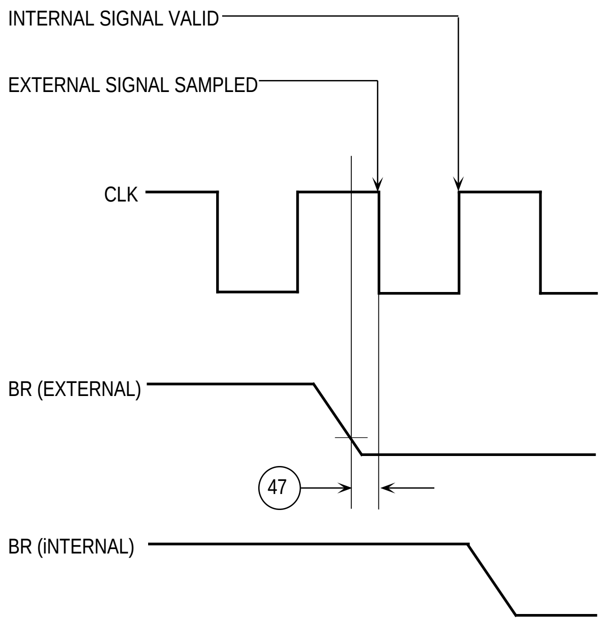

Figure 5-17. External Asynchronous Signal Synchronization

Bus arbitration control is implemented with a finite-state machine. State diagram (a) in Figure 5-18 applies to all processors using 3-wire bus arbitration and state diagram (b) applies to processors using 2-wire bus arbitration, in which $\overline{\text{BGACK}}$ is permanently negated internally or externally. The same finite-state machine is used, but it is effectively a two-state machine because $\overline{\text{BGACK}}$ is always negated.

In Figure 5-18, input signals R and A are the internally synchronized versions of $\overline{\text{BR}}$ and $\overline{\text{BGACK}}$. The $\overline{\text{BG}}$ output is shown as G, and the internal three-state control signal is shown as T. If T is true, the address, data, and control buses are placed in the high-impedance state when $\overline{\text{AS}}$ is negated. All signals are shown in positive logic (active high), regardless of their true active voltage level. State changes (valid outputs) occur on the next rising edge of the clock after the internal signal is valid.

A timing diagram of the bus arbitration sequence during a processor bus cycle is shown in Figure 5-19. The bus arbitration timing while the bus is inactive (e.g., the processor is performing internal operations for a multiply instruction) is shown in Figure 5-20.

When a bus request is made after the MPU has begun a bus cycle and before $\overline{\text{AS}}$ has been asserted (S0), the special sequence shown in Figure 5-21 applies. Instead of being asserted on the next rising edge of clock, $\overline{\text{BG}}$ is delayed until the second rising edge following its internal assertion.

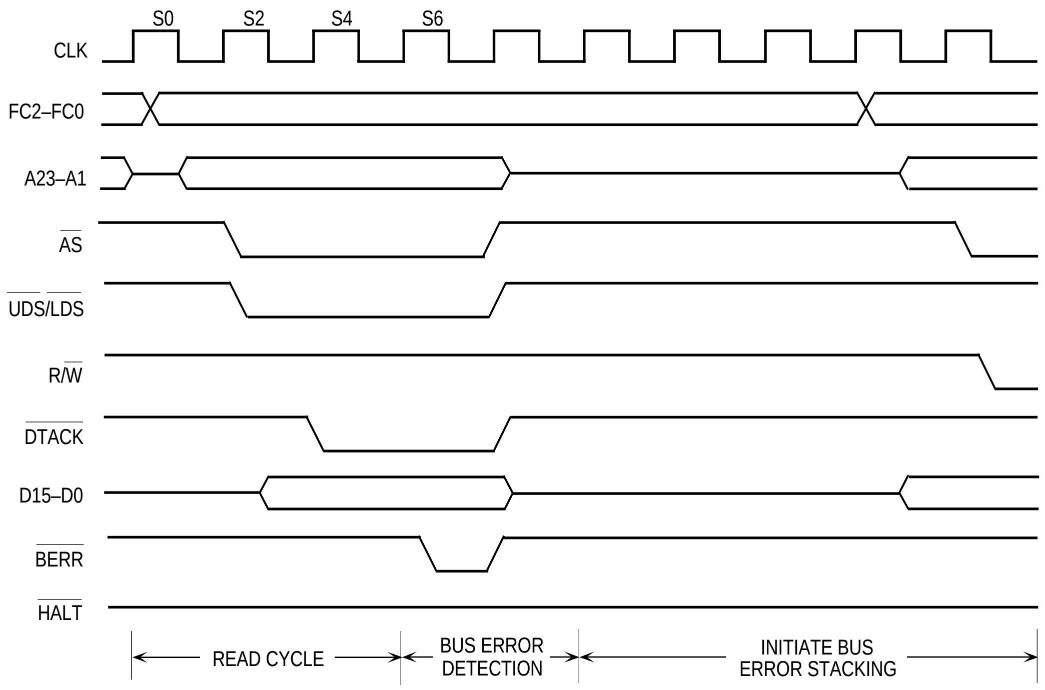

Figure 5-26. Delayed Bus Error Timing Diagram (MC68010)

After the aborted bus cycle is terminated and BERR is negated, the processor enters exception processing for the bus error exception. During the exception processing sequence, the following information is placed on the supervisor stack:

1. Status register
2. Program counter (two words, which may be up to five words past the instruction being executed)
3. Error information

The first two items are identical to the information stacked by any other exception. The error information differs for the MC68010. The MC68000, MC68HC000, MC68HC001, MC68EC000, and MC68008 stack bus error information to help determine and to correct the error. The MC68010 stacks the frame format and the vector offset followed by 22 words of internal register information. The return from exception (RTE) instruction restores the internal register information so that the MC68010 can continue execution of the instruction after the error handler routine completes.

After the processor has placed the required information on the stack, the bus error exception vector is read from vector table entry 2 (offset $08) and placed in the program counter. The processor resumes execution at the address in the vector, which is the first instruction in the bus error handler routine.

SECTION 6 EXCEPTION PROCESSING

This section describes operations of the processor outside the normal processing associated with the execution of instructions. The functions of the bits in the supervisor portion of the status register are described: the supervisor/user bit, the trace enable bit, and the interrupt priority mask. Finally, the sequence of memory references and actions taken by the processor for exception conditions are described in detail.

The processor is always in one of three processing states: normal, exception, or halted. The normal processing state is associated with instruction execution; the memory references are to fetch instructions and operands and to store results. A special case of the normal state is the stopped state, resulting from execution of a STOP instruction. In this state, no further memory references are made.

An additional, special case of the normal state is the loop mode of the MC68010, optionally entered when a test condition, decrement, and branch (DBcc) instruction is executed. In the loop mode, only operand fetches occur. See **Appendix A MC68010 Loop Mode Operation**.

The exception processing state is associated with interrupts, trap instructions, tracing, and other exceptional conditions. The exception may be internally generated by an instruction or by an unusual condition arising during the execution of an instruction. Externally, exception processing can be forced by an interrupt, by a bus error, or by a reset. Exception processing provides an efficient context switch so that the processor can handle unusual conditions.

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes the system is unusable and halts. Only an external reset can restart a halted processor. Note that a processor in the stopped state is not in the halted state, nor vice versa.

6.1 PRIVILEGE MODES

The processor operates in one of two levels of privilege: the supervisor mode or the user mode. The privilege mode determines which operations are legal. The mode is optionally used by an external memory management device to control and translate accesses. The mode is also used to choose between the supervisor stack pointer (SSP) and the user stack pointer (USP) in instruction references.

The privilege mode is a mechanism for providing security in a computer system. Programs should access only their own code and data areas and should be restricted from accessing information that they do not need and must not modify. The operating system executes in the supervisor mode, allowing it to access all resources required to perform the overhead tasks for the user mode programs. Most programs execute in user mode, in which the accesses are controlled and the effects on other parts of the system are limited.

6.1.1 Supervisor Mode

The supervisor mode has the higher level of privilege. The mode of the processor is determined by the S bit of the status register; if the S bit is set, the processor is in the supervisor mode. All instructions can be executed in the supervisor mode. The bus cycles generated by instructions executed in the supervisor mode are classified as supervisor references. While the processor is in the supervisor mode, those instructions that use either the system stack pointer implicitly or address register seven explicitly access the SSP.

6.1.2 User Mode

The user mode has the lower level of privilege. If the S bit of the status register is clear, the processor is executing instructions in the user mode.

Most instructions execute identically in either mode. However, some instructions having important system effects are designated privileged. For example, user programs are not permitted to execute the STOP instruction or the RESET instruction. To ensure that a user program cannot enter the supervisor mode except in a controlled manner, the instructions that modify the entire status register are privileged. To aid in debugging systems software, the move to user stack pointer (MOVE to USP) and move from user stack pointer (MOVE from USP) instructions are privileged.

NOTE

To implement virtual machine concepts in the MC68010, the move from status register (MOVE from SR), move to/from control register (MOVEC), and move alternate address space (MOVES) instructions are also privileged.

The bus cycles generated by an instruction executed in user mode are classified as user references. Classifying a bus cycle as a user reference allows an external memory management device to translate the addresses of and control access to protected portions of the address space. While the processor is in the user mode, those instructions that use either the system stack pointer implicitly or address register seven explicitly access the USP.

6.1.3 Privilege Mode Changes

Once the processor is in the user mode and executing instructions, only exception processing can change the privilege mode. During exception processing, the current state of the S bit of the status register is saved, and the S bit is set, putting the processor in the

6.4 RETURN FROM EXCEPTION (MC68010)

In addition to returning from any exception handler routine on the MC68010, the RTE instruction resumes the execution of a suspended instruction by returning to the normal processing state after restoring all of the temporary register and control information stored during a bus error. For the RTE instruction to execute properly, the stack must contain valid and accessible data. The RTE instruction checks for data validity in two ways. First, the format/offset word is checked for a valid stack format code. Second, if the format code indicates the long stack format, the validity of the long stack data is checked as it is loaded into the processor. In addition, the data is checked for accessibility when the processor starts reading the long data. Because of these checks, the RTE instruction executes as follows:

1. Determine the stack format. This step is the same for any stack format and consists of reading the status register, program counter, and format/offset word. If the format code indicates a short stack format, execution continues at the new program counter address. If the format code is not an MC68010-defined stack format code, exception processing starts for a format error.
2. Determine data validity. For a long-stack format, the MC68010 begins to read the remaining stack data, checking for validity of the data. The only word checked for validity is the first of the 16 internal information words (SP + 26) shown in Figure 5-8. This word contains a processor version number (in bits 10–13) and proprietary internal information that must match the version number of the MC68010 attempting to read the data. This validity check is used to ensure that the data is properly interpreted by the RTE instruction. If the version number is incorrect for this processor, the RTE instruction is aborted and exception processing begins for a format error exception. Since the stack pointer is not updated until the RTE instruction has successfully read all the stack data, a format error occurring at this point does not stack new data over the previous bus error stack information.
3. Determine data accessibility. If the long-stack data is valid, the MC68010 performs a read from the last word (SP + 56) of the long stack to determine data accessibility. If this read is terminated normally, the processor assumes that the remaining words on the stack frame are also accessible. If a bus error is signaled before or during this read, a bus error exception is taken. After this read, the processor must be able to load the remaining data without receiving a bus error; therefore, if a bus error occurs on any of the remaining stack reads, the error becomes a double bus fault, and the MC68010 enters the halted state.

Table 7-3. Move Word Instruction Execution Times

Source	Destination								
	Dn	An	(An)	(An)+	–(An)	(d_{16}, An)	(d_8, An, Xn)*	(xxx).W	(xxx).L
Dn	8(2/0)	8(2/0)	16(2/2)	16(2/2)	16(2/2)	24(4/2)	26(4/2)	24(4/2)	32(6/2)
An	8(2/0)	8(2/0)	16(2/2)	16(2/2)	16(2/2)	24(4/2)	26(4/2)	24(4/2)	32(6/2)
(An)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	32(6/2)	34(6/2)	32(6/2)	40(8/2)
(An)+	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	32(6/2)	34(6/2)	32(6/2)	40(8/2)
–(An)	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	34(6/2)	32(6/2)	34(6/2)	42(8/2)
(d_{16}, An)	24(6/0)	24(6/0)	32(6/2)	32(6/2)	32(6/2)	40(8/2)	42(8/2)	40(8/2)	48(10/2)
(d_8, An, Xn)*	26(6/0)	26(6/0)	34(6/2)	34(6/2)	34(6/2)	42(8/2)	44(8/2)	42(8/2)	50(10/2)
(xxx).W	24(6/0)	24(6/0)	32(6/2)	32(6/2)	32(6/2)	40(8/2)	42(8/2)	40(8/2)	48(10/2)
(xxx).L	32(8/0)	32(8/0)	40(8/2)	40(8/2)	40(8/2)	48(10/2)	50(10/2)	48(10/2)	56(12/2)
(d_{16}, PC)	24(6/0)	24(6/0)	32(6/2)	32(6/2)	32(6/2)	40(8/2)	42(8/2)	40(8/2)	48(10/2)
(d_8, PC, Xn)*	26(6/0)	26(6/0)	34(6/2)	34(6/2)	34(6/2)	42(8/2)	44(8/2)	42(8/2)	50(10/2)
#<data>	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	32(6/2)	34(6/2)	32(6/2)	40(8/2)

*The size of the index register (Xn) does not affect execution time.

Table 7-4. Move Long Instruction Execution Times

Source	Destination								
	Dn	An	(An)	(An)+	–(An)	(d_{16}, An)	(d_8, An, Xn)*	(xxx).W	(xxx).L
Dn	8(2/0)	8(2/0)	24(2/4)	24(2/4)	24(2/4)	32(4/4)	34(4/4)	32(4/4)	40(6/4)
An	8(2/0)	8(2/0)	24(2/4)	24(2/4)	24(2/4)	32(4/4)	34(4/4)	32(4/4)	40(6/4)
(An)	24(6/0)	24(6/0)	40(6/4)	40(6/4)	40(6/4)	48(8/4)	50(8/4)	48(8/4)	56(10/4)
(An)+	24(6/0)	24(6/0)	40(6/4)	40(6/4)	40(6/4)	48(8/4)	50(8/4)	48(8/4)	56(10/4)
–(An)	26(6/0)	26(6/0)	42(6/4)	42(6/4)	42(6/4)	50(8/4)	52(8/4)	50(8/4)	58(10/4)
(d_{16}, An)	32(8/0)	32(8/0)	48(8/4)	48(8/4)	48(8/4)	56(10/4)	58(10/4)	56(10/4)	64(12/4)
(d_8, An, Xn)*	34(8/0)	34(8/0)	50(8/4)	50(8/4)	50(8/4)	58(10/4)	60(10/4)	58(10/4)	66(12/4)
(xxx).W	32(8/0)	32(8/0)	48(8/4)	48(8/4)	48(8/4)	56(10/4)	58(10/4)	56(10/4)	64(12/4)
(xxx).L	40(10/0)	40(10/0)	56(10/4)	56(10/4)	56(10/4)	64(12/4)	66(12/4)	64(12/4)	72(14/4)
(d_{16}, PC)	32(8/0)	32(8/0)	48(8/4)	48(8/4)	48(8/4)	56(10/4)	58(10/4)	56(10/4)	64(12/4)
(d_8, PC, Xn)*	34(8/0)	34(8/0)	50(8/4)	50(8/4)	50(8/4)	58(10/4)	60(10/4)	58(10/4)	66(12/4)
#<data>	24(6/0)	24(6/0)	40(6/4)	40(6/4)	40(6/4)	48(8/4)	50(8/4)	48(8/4)	56(10/4)

*The size of the index register (Xn) does not affect execution time.

7.3 STANDARD INSTRUCTION EXECUTION TIMES

The numbers of clock periods shown in Table 7-5 indicate the times required to perform the operations, store the results, and read the next instruction. The total number of clock periods, the number of read cycles, and the number of write cycles are shown in the previously described format. The number of clock periods, the number of read cycles, and the number of write cycles, respectively, must be added to those of the effective address calculation where indicated by a plus sign (+).

Table 8-4. Standard Instruction Execution Times

Instruction	Size	op<ea>, An†	op<ea>, Dn	op Dn, <M>
ADD/ADDA	Byte, Word	**8**(1/0)+	**4**(1/0)+	**8**(1/1)+
	Long	**6**(1/0)+**	**6**(1/0)+**	**12**(1/2)+
AND	Byte, Word	—	**4**(1/0)+	**8**(1/1)+
	Long	—	**6**(1/0)+**	**12**(1/2)+
CMP/CMPA	Byte, Word	**6**(1/0)+	**4**(1/0)+	—
	Long	**6**(1/0)+	**6**(1/0)+	—
DIVS	—	—	**158**(1/0)+*	—
DIVU	—	—	**140**(1/0)+*	—
EOR	Byte, Word	—	**4**(1/0)***	**8**(1/1)+
	Long	—	**8**(1/0)***	**12**(1/2)+
MULS	—	—	**70**(1/0)+*	—
MULU	—	—	**70**(1/0)+*	—
OR	Byte, Word	—	**4**(1/0)+	**8**(1/1)+
	Long	—	**6**(1/0)+**	**12**(1/2)+
SUB	Byte, Word	**8**(1/0)+	**4**(1/0)+	**8**(1/1)+
	Long	**6**(1/0)+**	**6**(1/0)+**	**12**(1/2)+

+ Add effective address calculation time.
† Word or long only
* Indicates maximum basic value added to word effective address time
** The base time of six clock periods is increased to eight if the effective address mode is register direct or immediate (effective address time should also be added).
*** Only available effective address mode is data register direct.
DIVS, DIVU — The divide algorithm used by the MC68000 provides less than 10% difference between the best- and worst-case timings.
MULS, MULU — The multiply algorithm requires 38+2n clocks where n is defined as:
MULU: n = the number of ones in the <ea>
MULS: n=concatenate the <ea> with a zero as the LSB; n is the resultant number of 10 or 01 patterns in the 17-bit source; i.e., worst case happens when the source is $5555.

8.4 IMMEDIATE INSTRUCTION EXECUTION TIMES

The numbers of clock periods shown in Table 8-5 include the times to fetch immediate operands, perform the operations, store the results, and read the next operation. The total number of clock periods, the number of read cycles, and the number of write cycles are shown in the previously described format. The number of clock periods, the number of read cycles, and the number of write cycles, respectively, must be added to those of the effective address calculation where indicated by a plus sign (+).

In Table 8-5, the following notation applies:

#	—	Immediate operand
Dn	—	Data register operand
An	—	Address register operand
M	—	Memory operand

Table 10-1. Power Dissipation and Junction Temperature vs Temperature ($\theta JC=\theta JA$)

Package	T_A Range	θ_{JC} (°C/W)	P_D (W) @ T_A Min.	T_J (°C) @ T_A Min.	P_D (W) @ T_A Max.	T_J (°C) @ T_A Max.
L/LC	0°C to 70°C -40°C to 85°C 0°C to 85°C	15 15 15	1.5 1.7 1.5	23 -14 23	1.2 1.2 1.2	88 103 103
P	0°C to 70°C	15	1.5	23	1.2	88
R/RC	0°C to 70°C -40°C to 85°C 0°C to 85°C	15 15 15	1.5 1.7 1.5	23 -14 23	1.2 1.2 1.2	88 103 103
FN	0°C to 70°C	25	1.5	38	1.2	101

NOTE: Table does not include values for the MC68000 12F.
Does not apply to the MC68HC000, MC68HC001, and MC68EC000.

Table 10-2. Power Dissipation and Junction Temperature vs Temperature ($\theta JC \neq \theta JC$)

Package	T_A Range	θ_{JA} (°C/W)	P_D (W) @ T_A Min.	T_J (°C) @ T_A Min.	P_D (W) @ T_A Max.	T_J (°C) @ T_A Max.
L/LC	0°C to 70°C -40°C to 85°C 0°C to 85°C	30 30 30	1.5 1.7 1.5	23 -14 23	1.2 1.2 1.2	88 103 103
P	0°C to 70°C	30	1.5	23	1.2	88
R/RC	0°C to 70°C -40°C to 85°C 0°C to 85°C	33 33 33	1.5 1.7 1.5	23 -14 23	1.2 1.2 1.2	88 103 103
FN	0°C to 70°C	40	1.5	38	1.2	101

NOTE: Table does not include values for the MC68000 12F.
Does not apply to the MC68HC000, MC68HC001, and MC68EC000.

Values for thermal resistance presented in this manual, unless estimated, were derived using the procedure described in Motorola Reliability Report 7843 “Thermal Resistance Measurement Method for MC68XXX Microcomponent Devices”' and are provided for design purposes only. Thermal measurements are complex and dependent on procedure and setup. User-derived values for thermal resistance may differ.

10.4 CMOS CONSIDERATIONS

The MC68HC000, MC68HC001, and MC68EC000, with it significantly lower power consumption, has other considerations. The CMOS cell is basically composed of two complementary transistors (a P channel and an N channel), and only one transistor is turned on while the cell is in the steady state. The active P-channel transistor sources current when the output is a logic high and presents a high impedance when the output is logic low. Thus, the overall result is extremely low power consumption because no power is lost through the active P-channel transistor. Also, since only one transistor is turned on during the steady state, power consumption is determined by leakage currents.

Because the basic CMOS cell is composed of two complementary transistors, a virtual semiconductor controlled rectifier (SCR) may be formed when an input exceeds the supply voltage. The SCR that is formed by this high input causes the device to become latched in a mode that may result in excessive current drain and eventual destruction of the device. Although the MC68HC000 and MC68EC000 is implemented with input protection diodes, care should be exercised to ensure that the maximum input voltage specification is not exceeded. Some systems may require that the CMOS circuitry be isolated from voltage transients; other may require additional circuitry.

The MC68HC000 and MC68EC000, implemented in CMOS, is applicable to designs to which the following considerations are relevant:

1. The MC68HC000 and MC68EC000 completely satisfies the input/output drive requirements of CMOS logic devices.
2. The HCMOS MC68HC000 and MC68EC000 provides an order of magnitude reduction in power dissipation when compared to the HMOS MC68000. However, the MC68HC000 does not offer a "power-down" mode.

10.5 AC ELECTRICAL SPECIFICATION DEFINITIONS

The AC specifications presented consist of output delays, input setup and hold times, and signal skew times. All signals are specified relative to an appropriate edge of the clock and possibly to one or more other signals.

The measurement of the AC specifications is defined by the waveforms shown in Figure 10-2. To test the parameters guaranteed by Motorola, inputs must be driven to the voltage levels specified in the figure. Outputs are specified with minimum and/or maximum limits, as appropriate, and are measured as shown. Inputs are specified with minimum setup and hold times, and are measured as shown. Finally, the measurement for signal-to-signal specifications are shown.

NOTE

The testing levels used to verify conformance to the AC specifications does not affect the guaranteed DC operation of the device as specified in the DC electrical characteristics.

Num	Characteristic	8 MHz*		10 MHz*		12.5 MHz*		16.67 MHz 12F		16 MHz		20 MHz**		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
26[2]	Data-Out Valid to $\overline{\mathrm{DS}}$ Asserted (Write)	40	—	30	—	20	—	15	—	15	—	10	—	ns
27[5]	Data-In Valid to Clock Low (Setup Time on Read)	10	—	10	—	10	—	7	—	5	—	5	—	ns
27A[5]	Late $\overline{\mathrm{BERR}}$ Asserted to Clock Low (setup Time)	45	—	45	—	45	—	—	—	—	—	—	—	ns
28[2]	$\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$ Negated to $\overline{\mathrm{DTACK}}$ Negated (Asynchronous Hold)	0	240[11]	0	190	0	150	0	110	0	110	0	95	ns
28A	$\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$ Negated to Data-In High Impedance	—	187	—	150	—	120	—	110	—	110	—	95	ns
29	$\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$ Negated to Data-In Invalid (Hold Time on Read)	0	—	0	—	0	—	0	—	0	—	0	—	ns
29A	$\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$ Negated to Data-In High Impedance	—	187	—	150	—	120	—	90	—	90	—	75	ns
30	$\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$) Negated to $\overline{\mathrm{BERR}}$ Negated	0	—	0	—	0	—	0	—	0	—	0	—	ns
31[2,5]	$\overline{\mathrm{DTACK}}$ Asserted to Data-In Valid (Setup Time)	—	90	—	65	—	50	—	40	—	50	—	42	ns
32	$\overline{\mathrm{HALT}}$) and $\overline{\mathrm{RESET}}$ Input Transition Time	0	200	0	200	0	200	0	150	—	150	0	150	ns
33	Clock High to $\overline{\mathrm{BG}}$ Asserted	—	62	—	50	—	40	—	40	0	30	0	25	ns
34	Clock High to $\overline{\mathrm{BG}}$ Negated	—	62	—	50	—	40	—	40	0	30	0	25	ns
35	$\overline{\mathrm{BR}}$ Asserted to $\overline{\mathrm{BG}}$ Asserted	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
36[7]	$\overline{\mathrm{BR}}$ Negated to$\overline{\mathrm{BG}}$ Negated	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
37	$\overline{\mathrm{BGACK}}$ Asserted to $\overline{\mathrm{BG}}$ Negated	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clks
37A[8]	$\overline{\mathrm{BGACK}}$ Asserted to $\overline{\mathrm{BR}}$ Negated	20	1.5 Clks	20	1.5 Clks	20	1.5 Clks	10	1.5 Clks	10	1.5 Clks	10	1.5 Clks	ns
38	$\overline{\mathrm{BG}}$ Asserted to Control, Address, Data Bus High Impedance ($\overline{\mathrm{AS}}$ Negated)	—	80	—	70	—	60	—	50	—	50	—	42	ns
39	$\overline{\mathrm{BG}}$ Width Negated	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	clks
40	Clock Low to $\overline{\mathrm{VMA}}$ Asserted	—	70	—	70	—	70	—	50	—	50	—	40	ns
41	Clock Low to E Transition	—	55[12]	—	45	—	35	—	35	—	35	—	30	ns
42	E Output Rise and Fall Time	—	15	—	15	—	15	—	15	—	15	—	12	ns
43	$\overline{\mathrm{VMA}}$ Asserted to E High	200	—	150	—	90	—	80	—	80	—	60	—	ns
44	$\overline{\mathrm{AS}}$, $\overline{\mathrm{DS}}$ Negated to $\overline{\mathrm{VPA}}$ Negated	0	120	0	90	0	70	0	50	0	50	0	42	ns
45	E Low to Control, Address Bus Invalid (Address Hold Time)	30	—	10	—	10	—	10	—	10	—	10	—	ns
46	$\overline{\mathrm{BGACK}}$ Width Low	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	1.5	—	ns

Table A-1. MC68010 Loop Mode Instructions

Opcodes	Applicable Addressing Modes
MOVE [BWL]	(Ay) to (Ax) (Ay) to (Ax)+ (Ay) to –(Ax) (Ay)+ to (Ax) (Ay)+ to –(Ax) –(Ay) to (Ax) –(Ay) to (Ax)+ –(Ay) to –(Ax) Ry to (Ax) Ry to (Ax)+
ADD [BWL] AND [BWL] CMP [BWL] OR [BWL] SUB [BWL]	(Ay) to Dx (Ay)+ to Dx –(Ay) to Dx
ADDA [WL] CMPA [WL] SUBA [WL]	(Ay) to Ax –(Ay) to Ax (Ay)+ to Ax
ADD [BWL] AND [BWL] EOR [BWL] OR [BWL] SUB [BWL]	Dx to (Ay) Dx to (Ay)+ Dx to –(Ay)
ABCD [B] ADDX [BWL] SBCD [B] SUBX [BWL]	–(Ay) to –(Ax)
CMP [BWL]	(Ay)+ to (Ax)+
CLR [BWL] NEG [BWL] NEGX [BWL} NOT [BWL] TST [BWL] NBCD [B]	(Ay) (Ay)+ –(Ay)
ASL [W] ASR [W] LSL [W] LSR [W] ROL [W] ROR [W] ROXL [W] ROXR	(Ay) by #1 (Ay)+ by #1 –(Ay) by #1

NOTE: [B, W, or L] indicate an operand size of byte, word, or long word.